frobco
603 Mission Street
Santa Cruz, CA 95060

INSIDE THE ATARI VCS

HARDWARE

The central hardware functions of the VCS are performed by three LSI NMOS integrated circuits, the MPU (6507 Micro-Processor Unit), the PIA (6532 Peripheral Interface Adapter) and the TIA (Television Interface Adapter). The rest of the VCS components provide such functions as power regulation, noise filtering for the I/O lines, sound signal mixing and RF modulation.

MPU

The MPU comes from the 6502 family of microprocessors. The 6507 differs from the 6502 only in that some signal lines are not available allowing the IC to be placed in a 28 pin package instead of the usual 40 pin package. Among the missing pins are the address lines A15, A14 and A13. This restricts the address space to 8K bytes and causes all locations with the same lower 13 address bits to be mapped into the same physical location. For example, an access to locations 1234 (hex) has the same affect as an access to 3234, 5234, 7234, 9234, B234, D234, or F234.

The MPU is running on a 1.193 MHZ clock giving a processor cycle time of .8381 microseconds. There are no external interrupts connected to the MPU, however the TIA is connected to the ready line of the MPU and can cause the processor to pause for periods up to 64 microseconds. Although the external interrupt line is not connected, the program may use the BRK instruction to generate a soft interrupt.

PIA

The PIA provides three functions: dual 8 bit I/O ports, 128 bytes of RAM and a timer. One I/O port provides 4 lines to each of the two game controller connectors. These lines are usually used to read the 4 switches in the joysticks, however, they can also be programmed to be outputs. The RAM addresses start at 80 (hex) and go to FF. Due to lack of total address decoding, the RAM also shows up at 180-1FF providing a place for the system stack.

The timer in the PIA is used to count processor cycles in real time. A processor cycle is .8381 microseconds. When a number is written to the timer at one of its write locations (say 294-297 hex) it begins to decrement the 8 bit number. When it gets to zero it will just keep decrementing from there unless another number is written. The choice of write address in the block determines how many processor cycles it takes to cause one decrement. Using the first address (say 294 hex) sets the decrement to happen every processor cycle. The next address will set for one decrement every 8 processor cycles (6.705 microseconds). The third address (eg. 296 hex) will set for one

decrement every 64 processor cycles (53.64 microseconds), and the fourth address sets for one decrement every 1024 processor cycles (858.2 microseconds).

The program may look at the value of the timer at any time by reading an address (eg., 284 hex). The divide by 64 mode is very useful in VCS programming because a horizontal scan line takes 76 processor cycles, about 64 microseconds. This means one timer count will take a little less than one scan line.

The PIA has two eight bit ports. Port A is at memory address 280 hex and port B is at memory address 282 hex. The direction of the bits in these ports is programable via corresponding data direction registers. Write zeros in the data direction registers for inputs and ones in the data direction registers for outputs. Port A's data direction register is at 281 hex and port B's data direction register is at 283 hex. Writing a 00000001 into location 281 hex will program the upper 7 bits of port A to be inputs and the lower bit to be an output.

The PIA also has some interrupt options, but these are not supported by the 6507 MPU.

TIA

The TIA generates the video and audio signals to interface to the RF modulator which in turn produces a VHF signal on channel 2 or 3 to send to the TV set. An important part of this job is the production of the video synchronization pulses. This function is facilitated by circuits in the TIA, but the timing of the vertical pulses is controlled by the software in the cartridge.

The address space for the TIA is from 0 to 3F (hex). The lower section (0-2F) contain write only registers that control the video generator and sound system. The block from 30 to 3F is composed of read only registers that return collision informaton as well as the fire buttons on the sticks and paddle controls.

The cartridge interface connector has 24 pins. These pins provide the address bus lines A0 through A12 as well as the data bus lines D0 through D7 together with 5 volt power. Two very important signals that were NOT included are the processor clock and the processor read/write lines. The lack of these lines makes the design of peripherals for the VCS much more difficult than for most microcomputer devices.

MEMORY MAP

The TIA is selected for I/O if A12 is low and A7 is low, however, the TIA only uses A0 through A5 to decode internal addresses. Thus the 64 locations used by the TIA will appear in the following ranges (addresses in hex):

0-3F, 40-7F, 100-13F, 140-17F, 200-23F, 240-27F, 300-33F, 340-37F, 400-43F, 440-47F, 500-53F, 540-57F, 600-63F, 640-67F, 700-73F, 740-77F, 800-83F, 840-87F, 900-93F, 940-97F, A00-A3F, A40-A7F, B00-B3F, B40-B7F, C00-C3F, C40-C7F, D00-D3F, D40-D7F, E00-E3F, E40-E7F, F00-F3F, F40-F7F,

2000-203F, 2040-207F, 2100-213F, 2140-217F, 2200-223F, 2240-227F, 2300-233F, 2340-237F, 2400-243F, 2440-247F, 2500-253F, 2540-257F, 2600-263F, 2640-267F, 2700-273F, 2740-277F, 2800-283F, 2840-287F, 2900-293F, 2940-297F, 2A00-2A3F, 2A40-2A7F, 2B00-2B3F, 2B40-2B7F, 2C00-2C3F, 2C40-2C7F, 2D00-2D3F, 2D40-2D7F, 2E00-2E3F, 2E40-2E7F, 2F00-2F3F, 2F40-2F7F,

4000-403F, 4040-407F, 4100-413F, 4140-417F, 4200-423F, 4240-427F, 4300-433F, 4340-437F, 4400-443F, 4440-447F, 4500-453F, 4540-457F, 4600-463F, 4640-467F, 4700-473F, 4740-477F, 4800-483F, 4840-487F, 4900-493F, 4940-497F, 4A00-4A3F, 4A40-4A7F, 4B00-4B3F, 4B40-4B7F, 4C00-4C3F, 4C40-4C7F, 4D00-4D3F, 4D40-4D7F, 4E00-4E3F, 4E40-4E7F, 4F00-4F3F, 4F40-4F7F,

6000-603F, 6040-607F, 6100-613F, 6140-617F, 6200-623F, 6240-627F, 6300-633F, 6340-637F, 6400-643F, 6440-647F, 6500-653F, 6540-657F, 6600-663F, 6640-667F, 6700-673F, 6740-677F, 6800-683F, 6840-687F, 6900-693F, 6940-697F, 6A00-6A3F, 6A40-6A7F, 6B00-6B3F, 6B40-6B7F, 6C00-6C3F, 6C40-6C7F, 6D00-6D3F, 6D40-6D7F, 6E00-6E3F, 6E40-6E7F, 6F00-6F3F, 6F40-6F7F,

8000-803F, 8040-807F, 8100-813F, 8140-817F, 8200-823F, 8240-827F, 8300-833F, 8340-837F, 8400-843F, 8440-847F, 8500-853F, 8540-857F, 8600-863F, 8640-867F, 8700-873F, 8740-877F, 8800-883F, 8840-887F, 8900-893F, 8940-897F, 8A00-8A3F, 8A40-8A7F, 8B00-8B3F, 8B40-8B7F, 8C00-8C3F, 8C40-8C7F, 8D00-8D3F, 8D40-8D7F, 8E00-8E3F, 8E40-8E7F, 8F00-8F3F, 8F40-8F7F,

A000-A03F, A040-A07F, A100-A13F, A140-A17F, A200-A23F, A240-A27F, A300-A33F, A340-A37F, A400-A43F, A440-A47F, A500-A53F, A540-A57F, A600-A63F, A640-A67F, A700-A73F, A740-A77F, A800-A83F, A840-A87F, A900-A93F, A940-A97F, AA00-AA3F, AA40-AA7F, AB00-AB3F, AB40-AB7F, AC00-AC3F, AC40-AC7F, AD00-AD3F, AD40-AD7F, AE00-AE3F, AE40-AE7F, AF00-AF3F, AF40-AF7F,

C000-C03F, C040-C07F, C100-C13F, C140-C17F, C200-C23F,
C240-C27F, C300-C33F, C340-C37F, C400-C43F, C440-C47F,
C500-C53F, C540-C57F, C600-C63F, C640-C67F, C700-C73F,
C740-C77F, C800-C83F, C840-C87F, C900-C93F, C940-C97F,
CA00-CA3F, CA40-CA7F, CB00-CB3F, CB40-CB7F, CC00-CC3F,
CC40-CC7F, CD00-CD3F, CD40-CD7F, CE00-CE3F, CE40-CE7F,
CF00-CF3F, CF40-CF7F,

E000-E03F, E040-E07F, E100-E13F, E140-E17F, E200-E23F,
E240-E27F, E300-E33F, E340-E37F, E400-E43F, E440-E47F,
E500-E53F, E540-E57F, E600-E63F, E640-E67F, E700-E73F,
E740-E77F, E800-E83F, E840-E87F, E900-E93F, E940-E97F,
EA00-EA3F, EA40-EA7F, EB00-EB3F, EB40-EB7F, EC00-EC3F,
EC40-EC7F, ED00-ED3F, ED40-ED7F, EE00-EE3F, EE40-EE7F,
EF00-EF3F, EF40-EF7F.

The PIA is selected if A12 is low and A7 is high. Here only A0 through A6 are used internally, but A9 is used to select the bank of 128 bytes of RAM if high. The other locations are used for the I/O ports and timer control. The RAM bank will appear in the following ranges (addresses in hex):

80-FF, 180-1FF, 480-4FF, 580-5FF, 880-8FF, 980-9FF, C80-CFF,
D80-DFF,

2080-20FF, 2180-21FF, 2480-24FF, 2580-25FF, 2880-28FF,
2980-29FF, 2C80-2CFF, 2D80-2DFF,

4080-40FF, 4180-41FF, 4480-44FF, 4580-45FF, 4880-48FF,
4980-49FF, 4C80-4CFF, 4D80-4DFF,

6080-60FF, 6180-61FF, 6480-64FF, 6580-65FF, 6880-68FF,
6980-69FF, 6C80-6CFF, 6D80-6DFF,

8080-80FF, 8180-81FF, 8480-84FF, 8580-85FF, 8880-88FF,
8980-89FF, 8C80-8CFF, 8D80-8DFF,

A080-A0FF, A180-A1FF, A480-A4FF, A580-A5FF, A880-A8FF,
A980-A9FF, AC80-ACFF, AD80-ADFF,

C080-C0FF, C180-C1FF, C480-C4FF, C580-C5FF, C880-C8FF,
C980-C9FF, CC80-CCFF, CD80-CDFF,

E080-E0FF, E180-E1FF, E480-E4FF, E580-E5FF, E880-E8FF,
E980-E9FF, EC80-ECFF, ED80-EDFF.

The ports and timer locations will appear in the following ranges (again, addresses in hexadecimal):

280-2FF, 380-3FF, 680-6FF, 780-7FF, A80-AFF, B80-BFF, E80-EFF, F80-FFF,

2280-22FF, 2380-23FF, 2680-26FF, 2780-27FF, 2A80-2AFF, 2B80-2BFF, 2E80-2EFF, 2F80-2FFF,

4280-42FF, 4380-43FF, 4680-46FF, 4780-47FF, 4A80-4AFF, 4B80-4BFF, 4E80-4EFF, 4F80-4FFF,

6280-62FF, 6380-63FF, 6680-66FF, 6780-67FF, 6A80-6AFF, 6B80-6BFF, 6E80-6EFF, 6F80-6FFF,

8280-82FF, 8380-83FF, 8680-86FF, 8780-87FF, 8A80-8AFF, 8B80-8BFF, 8E80-8EFF, 8F80-8FFF,

A280-A2FF, A380-A3FF, A680-A6FF, A780-A7FF, AA80-AAFF, AB80-ABFF, AE80-AEFF, AF80-AFFF,

C280-C2FF, C380-C3FF, C680-C6FF, C780-C7FF, CA80-CAFF, CB80-CBFF, CE80-CEFF, CF80-CFFF,

E280-E2FF, E380-E3FF, E680-E6FF, E780-E7FF, EA80-EAFF, EB80-EBFF, EE80-EEFF, EF80-EFFF.

TIA CONTROL REGISTERS

VSYNCH (ADDRESS 0)

VSYNCH controls the polarity of the video signal. Writing a 2 to this location inverts the video to produce a vertical synch pulse. Writing a 0 will set it back to normal video operation. Most programs will only use this function to generate the vertical synch pulse, but it is possable to use this function for brief period within lines to produce special effects, however, because the horizontal synch pulses will be inverted, the screen may lose horizontal hold if this function is prolonged past one horizontal scan line.

VRESET (ADDRESS 1)

VRESET turns the video generator on and off. Writing a 2 to this location turns off the video. Storing a 0 will turn it back on again. This function is usually used during the generation of the vertical synch pulse to be sure no remaining video signal gets into the pulse. Often the video will not be turned back on right away giving the program a chance to "think" and thereby causing a black space at the top of the TV screen. VRESET may be

used at any time to force the screen to black without disturbing other control registers in the TIA.

The high order bit of VRESET controls the scan function for the game paddles. The TIA has two analog input lines each for the two game control connectors. Sending a code 80 (hex) to VRESET will hold these signals low. When the high order bit of VRESET is then cleared, these lines are free to charge up through varaible resistors in the paddles. Thus the program can "read" the value of a paddle by counting how much time it takes for the line to charge to threshold after the high order bit of VRESET has been set to zero.

LWAIT (ADDRESS 2)

Whenever the program writes anything to LWAIT the TIA pulls down the ready line on the MPU. This causes the MPU to hold the address bus and data bus while waiting for the TIA to let it go. The TIA will then let the MPU go about 5 microseconds before the start of the horizontal synch pulse. At this time the dot on the TV screen has just disappeared off the right side of the screen and is about to start over on the left side.

Nearly all timing for a VCS program is worked out in terms of stores to LWAIT. For the TV screen to lock in (NTSC American TV that is), there must be 262 horizontal scan lines between vertical synch pulses. The TIA keeps generating horizontal synch whether or not the program is using LWAIT. If the program is not using LWAIT to synchronize and count scan lines, then it must use the timer in the PIA or count all processor cycles to know when to generate the vertical synch pulse.

LWAIT also can change the internal state of the TIA. Sometimes a program will do some of the lines of a screen on absolute cycle counting insted of using LWAIT in order to achieve some special effects.

VID03 (ADDRESS 3)

We have never seen any game use this location. If a program writes to this location it causes a tempory change in the timing of the horizontal synch pulses. This may be of use in programs for PAL or SECAM systems. This feature is being studied at this time and will be further explained in an upcoming application note.

P1VMODE (address 4)

This register controls the video mode for the player1 sprite and the player1 shot sprite. A sprite is a group of video picture elements (pixels) that form some pattern on the screen and can be moved around on a screen without disturbing the pattern or any background that may be temporarily obscured. The VCS has five sprites: player1, player2, shot1, shot2 and shot3. The player sprites are named such because they were used in the early games to hold the pictures of the two players of the game. The players usually shoot at each other, so these one bit sprites are called "shots." The shots are only one pixel, once enabled they appear on all successive scan lines until turned off by the software.

The player sprites hold 8 bits. These bits are written by the software into specific TIA addresses (see P1IMAG below). Once written the image will be displayed on all following scan lines until a new image is written. The sprite can be turned off by setting the image to zero (all clear).

The shot sprites are only one pixel. They therefore do not need image registers, but they do have enable registers. When a shot is enabled, it appears on all scan lines until disabled.

P1VMODE controls the horizontal size of the pixels used to display the player1 image and shot, as well as the number of times to repeat the image or shot along the scan lines. A horizontal scan line is made up of 160 High Resolution Dots (HRD's). Each pixel of a player image may be 1, 2, or 4 HRD's wide depending on the value of the associated VMODE register.

Bits 5 and 4 of P1VMODE control the horizontal width of the shot associated with player1. The shot may be 1, 2, 4, or 8 HRD's wide.

VMODE CODES

ØØ = One player image at 1 HRD/pixel
One shot at 1 HRD/pixel

Ø1 = Two player images separated by one player width at 1 HRD/pixel
Two shot separated by one player width at 1 HRD/pixel

Ø2 = Two player images separated by two player widths at 1 HRD/pixel
Two shot separated by two player widths at 1 HRD/pixel

Ø3 = Three player images separated by one player width each at 1 HRD/pixel
Three shot separated by one player width each at 1 HRD/pixel

Ø4 = Two player images separated by five player widths at 1 HRD/pixel
Two shot separated by five player widths at 1 HRD/pixel

Ø5 = One player image at 2 HRD/pixel
One shot at 1 HRD/pixel

Ø6 = Three player images separated by two player widths at 1 HRD/pixel
Three shot separated by two player widths at 1 HRD/pixel

Ø7 = One player image at 4 HRD/pixel

1X = Set shot at 2 HRD/pixel

2X = Set shot at 4 HRD/pixel

3X = Set shot at 8 HRD/pixel

P2VMODE (ADDRESS 5)

See P1VMODE but substitute player2 for player 1 and player2 shot for playerl shot.

P1COLOR (ADDRESS 6)

The color register associated with the playerl image and shot. This color may also be used for parts of the object fields if the proper bits are set in FVMODE. The high order 4 bits of the color value set the chrominance (color shade) and the lower 4 bits set the luminance (color brightness). Here are some examples:

ØØ = Black
Ø8 = Light grey
ØF = White
18 = Bright yellow
28 = Orange yellow
38 = Orange
44 = Red
48 = Pink
58 = Light magenta
68 = Violet
78 = Blue Violet
88 = Blue
98 = Blue green
A8 = Blue green
B8 = Green
C8 = Green yellow
D8 = Light yellow
E8 = Yellow
F8 = Orange yellow

P2COLOR (ADDRESS 7)

Same as above but for the player2 image and shot.

OCOLOR (ADDRESS 8)

This is the object field color register. The format is the same as P1COLOR and P2COLOR. This color is always used by shot3 and may be used by the objects (see FVMODE).

BCOLOR (ADDRESS 9)

Register 9 holds the color value for the screen background. Again the format is the same as the player color registers. If any image has the same color value as the BCOLOR, it will not show up on the screen unless it passes in front of (has higher priority than) another image with a different color value.

FVMODE (ADDRESS ØA HEX)

This is the field video mode register. It controls a great number of things. One of the things it controls is whether the low resolution graphics are shifted out left to right or right to left on the second half of the screen. As discussed earlier the low resolution graphics are displayed from three other registers. Two of these registers hold eight bits of image and one of the registers holds 4 bits of image. So the total is twenty bits. If the low order bit of the FVMODE register is set then the right half of the screen will contain a copy of the low-res graphics except for the fact that they will be flipped over creating a screen which has bilateral symmetry. Here are some codes for the FVMODE register.

ØØ = Low resolution field objects are controlled by their own color registers, and there is no flip. The foreground to background display priority of the sprites and objects are from highest to lowest: player1, player2, the objects together with shot3 followed by the background.

Ø1 = Flips over the right side of the low resolution graphics.

Ø2 = No flip over but the player1 color and priority is associated with the low resolution object field on the left half of the screen. player2 color and priority are associated with the low resolution object field on the right half of the screen.

Ø3 = A combination of 1 and 2 above.

Ø4 = Object fields are associated with their own color except that the foreground to background display priority is the objects and shot3, player1, player2 followd by the background. Shot3 gets its color from object color so it is now displayed in front of the player1, player2, shot1 and shot2.

Ø5 = Flips right side objects.

Ø6 = No flip. Looks like a code 4.

Ø7 = Looks like a code 5.

Ø8 = Looks like code Ø.

All codes up to 1Ø hex are repeats of the above.

1X = Causes display of shot3 to be 2 HRDs.

2X = Causes display of shot3 to be 4 HRDs.

3X = Causes display of shot3 to be 8 HRDs.

P1FLIP (ADDRESS ØB HEX)

Writing a value of 8 to this location causes a left-right flip of the player1 image.

P2FLIP (ADDRESS ØC HEX)

Writing a value of 8 to this location causes a left-right flip of the player2 image.

FLDAIM (ADDRESS ØD HEX)

This register holds the most significant 4 bits (part A) of the 2Ø bit object field register. These bits are in the lower 4 bits of FLDAIM.

FLDBIM (ADDRESS ØE HEX)

This register holds bits 8 through 15 (part B) of the 2Ø bit object field register.

FLDCIM (ADDRESS ØF HEX)

This register holds bits Ø through 7 (part C) of the 2Ø bit object field register.

P1HRES (ADDRESS 1Ø HEX)

Writing to P1HRES causes the absolute horizontal position of the player1 sprite to be set to a place on the scan line acording to how many processor cycles occur between the start of the scan line and the store to P1HRES. If the current scan position is to the left of the visible screen, then the sprite is set to the left edge. If the current scan position is on the visible screen, then the sprite is set to that position.

P2HRES (ADDRESS 11 HEX)

Write anything to P2HRES to set the absolute horizontal position of the player2 image as in P1HRES.

S1HRES (ADDRESS 12 HEX)

Write anything to S1HRES to set the absolute horizontal position of the player1 shot as in P1HRES.

S2HRES (ADDRESS 13 HEX)

Write anything to S2HRES to set the absolute horizontal position of the player2 shot as in P1HRES.

S3HRES (ADDRESS 14 HEX)

Write anything to S3HRES to set the absolute horizontal position of shot3 as in P1HRES.

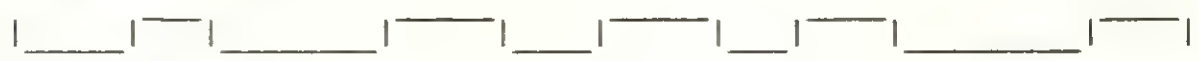
SND1MD (ADDRESS 15 HEX)

A program sets the sound mode for audio generator 1 by writing a 5 bit code to location SND1MD. These codes select the waveform and frequency range of the sound produced by the audio generator.

Waveform type A:

Waveform type B:

Waveform type C:



(type C continued)

CODE (hex)	DESCRIPTION
ØØ	No oscillation
Ø1	Type A wave that repeats with period equal to (SND1TN+1)*477.7 microseconds.
Ø2	Type A wave that repeats with period equal to (SND1TN+1)*7.4Ø4 milliseconds.
Ø3	Mix of waveforms, nonrandom noise.
Ø4	Square wave at a frequency of 15.6996/(SND1TN+1) Khz.
Ø5	Same as Ø4.
Ø6	Square wave at a frequency of 1.Ø129/(SND1TN+1) Khz.
Ø7	Type B wave that repeats with period equal to (SND1TN+1)*987.2 microseconds.
Ø8	Another mix of waveforms, nonrandom noise.
Ø9	Inverted type B wave that repeats with period equal to (SND1TN+1)*987.2 microseconds.
ØA	Same as Ø6
ØB	No signal
ØC	Square wave at a frequency of 5.2331/(SND1TN+1) Khz.
ØD	Same as ØC
ØE	No signal
ØF	Type C wave that repeats with period equal to (SND1TN+1)*2,957 microseconds.

SND2MD (ADDRESS 16 HEX)

This provides the same function as SND1MD except for audio channel 2.

SND1TN (ADDRESS 17 HEX)

The SND1TN register sets the tone or base frequency of the sound generated by audio channel 1. The range of the frequency is set by the mode register while the lower 5 bits of SND1TN select a frequency within that range. Here are some examples for simple square wave modes:

	(frequency in hertz)		
CODE (hex)	MODE=04	MODE=0C	MODE=06
00	15,699.6	5,233.2	1,012.9
01	7,849.8	2,616.6	506.5
02	5,233.2	1,744.4	337.6
03	3,924.9	1,308.3	253.2
04	3,139.9	1,046.6	202.6
05	2,616.6	872.2	168.8
06	2,242.8	747.6	144.7
07	1,962.5	654.1	126.6
08	1,744.4	581.5	112.5
09	1,570.0	523.3	101.3
0A	1,427.2	475.7	92.1
0B	1,308.3	436.1	84.4
0C	1,207.7	402.5	77.9
0D	1,121.4	373.8	72.4
0E	1,046.6	346.9	67.5
0F	981.2	327.1	63.3
10	923.5	307.8	59.6
11	872.2	290.7	56.3
12	826.3	275.4	53.3
13	785.0	261.7	50.7
14	747.6	249.2	48.2
15	713.6	237.9	46.0
16	682.6	227.5	44.0
17	654.2	218.0	42.2
18	628.0	209.3	40.5
19	603.8	201.3	39.0
1A	581.5	193.8	37.5
1B	560.7	186.9	36.2
1C	541.4	180.5	34.9
1D	523.3	174.4	33.8
1E	506.4	168.8	32.7
1F	490.6	163.5	31.7

Codes in Terms of Musical Notes:

NOTE	FREQ	SND1MD	SND1TN	ERROR
A6	1,760.0	04	08	-0.9%
G5#	1,661.2	OUT OF RANGE		
G5	1,568.0	04	09	+0.1%
F5#	1,480.0	OUT OF RANGE		
F5	1,396.9	04	0A	+2.1%
E5	1,318.5	04	0B	-0.8%
D5#	1,244.5	04	0C	+2.6%
D5	1,174.7	OUT OF RANGE		
C5#	1,108.7	04	0D	+1.1%
C5	1,046.5	04	0E	0.0%
B5	987.8	04	0F	-0.7%
A5#	932.3	04	10	-1.2%
A5	880.0	04	11	-0.9%
G4#	830.6	04	12	-0.5%
G4	784.0	04	13	+0.1%
F4#	740.0	04	14	+1.0%
F4	698.5	04	15	+2.2%
E4	659.3	04	17	-0.8%
D4#	622.3	04	18	+0.9%
D4	587.3	04	1A	-1.0%
C4#	554.4	04	1C	-2.3%
C4	523.3	04	1D	0.0%
B4	493.9	04	1F	-0.7%
A4#	466.2	0C	0A	+2.0%
A4	440.0	0C	0B	-0.9%
G3#	415.3	OUT OF RANGE		
G3	392.0	0C	0C	+2.6%
F3#	370.0	0C	0D	+1.0%
F3	349.2	0C	0E	-0.1%
E3	329.6	0C	0F	-0.8%
D3#	311.1	0C	10	-1.1%
D3	293.7	0C	11	-1.0%
C3#	277.2	0C	12	-0.6%
C3	261.6	0C	13	0.0%
B3	246.9	0C	14	+1.0%
A3#	233.1	0C	15	+2.1%
A3	220.0	0C	17	-0.9%
G2#	207.7	0C	18	+0.8%
G2	196.0	0C	1A	-1.1%
F2#	185.0	0C	1B	+1.0%
F2	174.6	0C	1D	-0.1%
E2	164.8	0C	1F	-0.8%
D2#	155.6	OUT OF RANGE		
D2	146.8	06	06	-1.4%
C2#	138.6	OUT OF RANGE		
C2	130.8	OUT OF RANGE		
B2	123.5	06	07	+2.5%
A2#	116.5	OUT OF RANGE		
A2	110.0	06	08	+2.3%
G1#	103.8	06	09	-2.4%

G1	98.00	OUT OF RANGE		
F1#	92.50	06	0A	-0.5%
F1	87.31	OUT OF RANGE		
E1	82.41	06	0B	+2.4%
D1#	77.82	06	0C	+0.1%
D1	73.42	06	0D	-1.5%
C1#	69.30	06	0E	-2.6%
C1	65.41	OUT OF RANGE		
B1	61.74	06	0F	+2.5%
A1#	58.27	06	10	+2.2%
A1	55.00	06	11	+2.3%
G0#	51.91	06	13	-2.4%
G0	49.00	06	14	-1.6%

SND2TN (ADDRESS 18 HEX)

This register provides the same function as SND1TN but as applied to audio channel 2.

SND1AM (ADDRESS 19 HEX)

The SND1AM is a four bit register that sets the amplitude or loudness of the signal produced by the channel 1 audio generator. A value of 00 produces no sound and a value of 0F (hex) is full volume.

SND2AM (ADDRESS 1A HEX)

Again this is the corresponding amplitude register for audio channel 2.

P1IMAG (ADDRESS 1B HEX)

P1IMAG holds the 8 bit video image for the player1 sprite.

P2IMAG (ADDRESS 1C HEX)

P2IMAG holds the 8 bit video image for the player2 sprite.

P1SHOT (ADDRESS 1D HEX)

Writing a 2 to location P1SHOT will enable the player1 shot sprite to be shown on the next scan line and all scan lines thereafter until a 0 is written into bit 1. For the sprite to be visible it also must be enabled by a 0 in bit 1 of S1CONT below. P1SHOT allows the shot to be turned on and off without changing its absolute horizontal position.

P2SHOT (ADDRESS 1E HEX)

This register is the same as P1SHOT, but works with shot2.

S3SHOT (ADDRESS 1F HEX)

This register is the same as P1SHOT, but works with shot3.

P1HORZ (ADDRESS 20 HEX)

P1HORZ is the horizontal increment value for the player1 sprite. Only the upper 4 bits of this register are used. These bits represent a signed 4 bit number. See below for more discussion of the use of the horizontal increment value.

P2HORZ (ADDRESS 21 HEX)

This is the same as P1HORZ, but used for the player2 sprite.

S1HORZ (ADDRESS 22 HEX)

This is the same as P1HORZ, but used for shot1.

S2HORZ (ADDRESS 23 HEX)

This is the same as P1HORZ, but used for shot2.

S3HORZ (ADDRESS 24 HEX)

This is the same as P1HORZ, but used for shot3.

P1DELAY (ADDRESS 25 HEX)

If a one is written to bit zero of this register it causes the image written to P1IMAG to be delayed until next line. This seems to be useful in "stacking" parts of images ahead of when they need to appear.

P2DELAY (ADDRESS 26 HEX)

This functions as does P1DELAY, but with respect to the player2 sprite.

VID27 (ADDRESS 27 HEX)

The function of this register is currently unknown.

S1CONT (ADDRESS 28 HEX)

Writing a 2 to this register turns off shot1. Writing a Ø to this register turns shot1 back on again, but the absolute horizontal position of shot1 will be loaded from the absolute horizontal position of the player1 sprite. This causes shot1 to be reset back to the player1 sprite, and is therefore very useful in setting up before shooting.

S2CONT (ADDRESS 29 HEX)

This register operates in the same manner as S1CONT only it is for shot2.

HZSCRL (ADDRESS 2A HEX)

HZSCRL is another strobe register. (what is written to it does not matter, but rather when is very important) If you write to it at the beginning of a line, then it takes the value of each of the sprites horizontal increment values and adds them to each of the sprites current absolute horizontal position. These increments may be negitive so motion can be right or left. An artifact of this proceedure is a tempory shutdown of the video signal while the process is taking place. This is seen as a short black line segment at the left edge of the screen.

If you write any value to this register at any time other than at the beginning of the line then all of the sprites move to the right 5 HRD's. When a sprite reaches the right edge of the screen it wraps around to the left side of the screen.

NOINC (ADDRESS 2B HEX)

Writing anything to this location sets the horizontal increment values of all the sprites to zero. This is very usefull when a program is going to hit HZSCRL but does not want the horizontal positions of the sprites disturbed.

COLRES (ADDRESS 2C HEX)

Writing any value to this register resets all of the latches for the collision detection registers.

VID2D (ADDRESS 2D HEX)

The function of this register is currently unknown.

VID2E (ADDRESS 2E HEX)

The function of this register is currently unknown.

VID2F (ADDRESS 2F HEX)

The function of this register is currently unknown.

This ends the section of write only registers. Reading any of the registers from Ø to 2F hex will return you the collision detection register corresponding to the lower 4 bits of the next set of registers 3Ø to 3F hex. As an example, reading locations Ø2, 12 or 22 will give you the same register as reading 32.

PS1COL (ADDRESS 3Ø HEX)

This is the shotl collision register. Bit 7 is set if player2 collides with shotl. Bit Ø is set if playerl collides with shotl.

PS2COL (ADDRESS 31 HEX)

This is the shot2 collision register. Bit 7 is set if playerl collides with shot2. Bit Ø is set if player2 collides with shot2.

P1OCOL (ADDRESS 32 HEX)

This is the player 1 to object field collision register. Bit 7 is set if playerl collides with any of the object fields.
BIt Ø is set if playerl collides with shot3.
P2OCOL (ADDRESS 33 HEX)

Bit 7 is set if player2 collides with any of the object fields.
Bit Ø is set if player2 collides with shot3.

S1OCOL (ADDRESS 34 HEX)

Bit 7 is set if shotl collides with any of the object fields.
Bit Ø is set if shotl collides with shot3.

S2OCOL(ADDRESS 35 HEX)

Bit 7 is set if shot2 collides with any of the object fields. Bit Ø is set if shot2 collides with shot3.

S3OCOL (ADDRESS 36 HEX)

Bit 7 is set if shot3 collides with any of the object fields.

PPCOL (ADDRESS 37 HEX)

Bit 7 is set if player1 collides with player2.

PDL5L (ADDRESS 38 HEX)

Bit 7 is set if pin 5 of the left game connector has gone above 2 volts subsequent to the last time the high order bit of VRESET has been turned off. (i.e. the paddle line has charged up)

PDL9L (ADDRESS 39 HEX)

This is the same as PDL5L, except it looks at the other paddle comming in on pin 9.

PDL5R (ADDRESS 3A HEX)

This is the same as PDL5L, except that the signal is comming from pin 5 on the right game controller connector.

PDL9R (ADDRESS 3B HEX)

This is the same as PDL9L, except that the signal is comming from pin 9 on the right game controller connector.

LFTFR (ADDRESS 3C HEX)

Pressing the fire button on left joy stick causes bit 7 to go low.

RGHFR (ADDRESS 3D HEX)

Pressing the fire button on the right joy stick causes bit 7 to go low.

VID3E (ADDRESS 3E HEX)

The function of this register is currently unknown.

VID3F (ADDRESS 3F HEX)

The function of this register is currently unknown.

READING SWITCHES AND BUTTONS

The VCS I/O interface through the PIA allows the software to read the switches on the console and to communicate with the game controllers. The connection to the console switches is very straightforward requiring only that the progrmmer set port B of the PIA to all inputs (i.e. write Ø to loc 283 hex). After this has been done the software can get the switch values by reading location 282 hex. The following table shows the bit assignment for the value returned:

B7	B6	B5	B4	B3	B2	B1	BØ
1 RIGHT DIF A	LEFT DIF A	NOT USED	NOT USED	COLOR	NOT USED		
Ø RIGHT DIF B	LEFT DIF B	NOT USED	NOT USED	B & W	NOT USED	GAME SELECT	GAME RESET

It is interesting to note here that many users of the VCS believe that the B&W/COLOR and GAME RESET switches are connected to the system hardware and perform their functions through hardware modes. This is not the case, but rather these switches are just inputs to the software and it is up to the programmer to assign some function to them. Some of the new games are using the switches for game play inputs that have nothing to do with their uses printed on the front pannel.

The eight lines of PIA port A are split into two groups of four and are connected to pins of the game controller connectors. The mapping is as follows:

B7	B6	B5	B4	B3	B2	B1	BØ
LEFT PIN 4	LEFT PIN 3	LEFT PIN 2	LEFT PIN 1	RIGHT PIN 4	RIGHT PIN 3	RIGHT PIN 2	RIGHT PIN 1

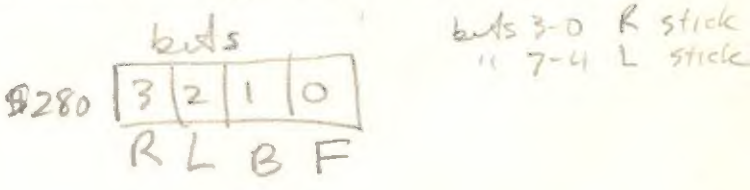
These lines can be programmed as either inputs or outputs on a line-by-line basis. When used as inputs they "float high", this means they will read as 1's if nothing is connected to them (e.g. if no connectors are pluged in and all lines are set for input, then a program will read an FF hex when accessing loc 280 hex).

JOY STICKS

The joy stick controllers are actually a set of five switches. They have one switch each for up, down, left and right as well as one for the fire button. The direction switches are wired to port A of the PIA, however the fire button goes into the TIA and controls a bit in one of the TIA read registers (see above). When two joy sticks are connected loc 280 hex returns the left stick direction in the upper four bits and the right stick position in the lower four bits. The closing of a switch causes the line to be connected to the zero volt reference (pin 8 of the connector) which causes the corresponding bit in port A of the PIA to become a zero. The bits are not latched so when the switch opens again the bit will go back to the one state.

Here are the four bit codes in hex that represent the stick positions using the face-of-the-clock terminology where TOP on the stick is at 12:ØØ.

No Action	= F (hex)
12:ØØ	= E
1:3Ø	= 6
3:ØØ	= 7
4:3Ø	= 5
6:ØØ	= D
7:3Ø	= 9
9:ØØ	= B
10:3Ø	= A



GAME PADDLES

The game paddles are constructed so that two paddles connect to one game controller socket. Thus it is possible to make a four player game that is controled by the paddles. To allow for this it was necessary to make the fire button on the paddles come in on a different line than used by the joy sticks so that four lines could be input at once. To acomplish this the game paddle fire buttons are connected to the top two bits of the

corresponding four bit group of port A. As with the joy sticks the bit will become zero while the game paddle fire button is held down.

The reading of the knob setting of the paddle comes in on the analog input lines. See the section on VRESET and the PDL registers of the TIA given above.

KEYBOARD CONTROLERS

The keyboard controllers are sets of twelve buttons that connect one each to the game controller sockets. Atari also came out with the Video Touch Pad to be used with Star Raiders. These devices are electrically the same, but the VTP gives much less wair and tair on your fingers.

Internally the keyboard controller switches cause connections to be made between row and column lines. Rows 1, 2, 3 and 4 (starting from the top) are connected to game controller pins 1, 2, 3 and 4. Columns 1, 2 and 3 (left to right) are connected to pins 5, 9, and 6. Lines 5 and 9 (the analog input lines) also have 4700 ohm resistors connected to +5 volts at pin 7. Pin 6 is the joy stick fire button input.

In order for the software to detect which button is pressed it is necessary to program the row select bits of PIA port A to be outputs, and then sending a zero to only one row select line at a time. Each time one of the row select lines goes low it will cause the three column lines to reflect the states of the three buttons of the selected row. If pressed the column line will go to zero. This is reflected directly in the TIA register for the fire button, and the other two columns (pins 5 and 9) come in on the high bits of the PDL registers of the TIA.

ABSOLUTE HORIZONTAL POSITIONING

Horizontal control is always the most difficult part of any VCS program. This is because the machine does not have a register that you can write in order to set the absolute horizontal position of a sprite. Furthermore, because the screen must always be recreated on the fly, you do not have time to do very much computation. This time shortage is really the worst part of the problem so let us look as some of its ramifications.

The TIA is running on a clock with frequency of 3.579545 MHZ. Every cycle of this clock the TIA shifts out one High Resolution Dot (HRD). The MPU is clocked by a signal that is 1/3 the frequency of the TIA clock. This means that for every instruction cycle the of the MPU the TIA has put up three HRD. A simple instruction like STA P1IMAG takes three MPU cycles and therefore spans nine HRD.

Do to horizontal blanking only 160 of the 228 HRD/line show up on the screen. When the program does a store to LWAIT the system waits until the scan is off the end of the current line and is 69 HRD from the visible left edge of the screen. A scan line takes 76 MPU cycles so therefore starting after a store to LWAIT you have 22.67 cycles until visible and 53.33 cycles across the screen.

The most common method of setting the absolute horizontal position of a sprite is as follows:

1. Get to the start of a line.
2. Wait enough MPU cycles to get within 8 HRD of the desired horizontal position.
3. Store to P1HRES (or the corresponding register for a different sprite).
4. Store a value into the upper 4 bits of the increment register to adjust position up to + or - 8 HRD.
5. At the start of the next line store to HZSCRL which will cause the increment to be added to the position set by P1HRES.

It is typical for a program to use a five cycle loop to do the timing wait before P1HRES. This gives 15 HRD/loop so screen positions will have 15 HRD between stops. Then the four bit fine increment value can reach all the points inbetween.

For more information and examples see the listings of the screen2 section of the EXPLORER program supplied with the Frobco software.

END OF DOCUMENT